A TOOLKIT FOR MANAGING MULTI-PROTOCOL INTERCONNECTIONS

R. State, E. Nataf O. Festor
INRIA-LORIA
615, Rue de Jardin Botanique – B.P.101
Villers-les-Nancy, 54600
France
{state, nataf, festor}@loria.fr

Abstract: *Within the last few years, many new technologies have emerged, appearing to be promising to the domain of equipment, network and service management. Often shown as competitors to TMN principles and underlying models, these approaches have been used sparsely in various projects.*

In this paper, we present the use of one of these technologies, namely Java, to implement a network information model which captures the essence of multi-protocol interconnections. We apply the developed concepts for the real-world scenarios of ATM and Frame Relay interconnection. We show how the TMN concepts stay valid and how the existing and extended information models can be reused and implemented according to the TMN principles in a full Java environment.

KEYWORDS: Network Information Model, Tina-C, ATM, Frame Relay, Java, UML

1. INTRODUCTION

Multiple technologies are the building blocks for modern computer technologies. Interconnecting and achieving interoperability at a service level between users located on different types of networks is of a particular challenge from a management point of view. A promising approach to manage such a heterogeneous network consisting of several interconnected transport networks is based on the paradigm of integrated network management.

In order to perform the management, a good representation of a network is needed. Such a representation is given by a network information model which can model the resources of one network. These resources can be physical interconnection links, cross-connections in an ATM switch, QoS resources and more complex entities that we have to deal with in modern computer networks. A network information model will allow us to represent in a top down approach a particular network. Using an object oriented modelling approach, we are able to manage the tremendous complexity which is inherent to systems containing such a huge amount of components. Several models for this purpose exist today and most of them, whilst inspired by OSI based specifications, are defined using the Unified Modelling Language (UML) [1] notation.

In this paper we focus on the implementation of such a model into a Java based management platform which tries to combine both OSI views of management information as well as more advanced interfaces to navigate among such a large model. The case study considered here concerns the interconnection of Frame Relay and ATM technologies. The interoperability of these technologies is important in building large scale enterprise networks, since it permits the reuse of already existent Frame Relay infrastructure with the more recent ATM. From a management point of view, we were challenged in our work, to combine UML based notations and development tools with both OSI and non OSI-based management architectures.

The paper is organised as follows. We will start with a brief introduction to the interoperability of ATM and Frame Relay. Next, we will introduce a network information model which can model the types of interconnections defined by the standardisation bodies. Based on this model, we will describe the Java architecture over which this model has been implemented. Finally, a summary of the presented work is given and future work is outlined.

2. CASE STUDY

In this section we will briefly review the two most used interconnection types between ATM and Frame Relay taken as the case study for our management implementation.

The two international organizations involved in developing standards for the ATM and Frame Relay technologies are the ATM forum and the Frame Relay forum. Both released a joint standard for the interconnection of ATM and Frame Relay, which is specified in the FRF5 [2] and FRF8 [3] documents. These documents contain the specifications for the two most important types of interconnection needed to interoperate between ATM and Frame Relay.

2.1. ATM/FRAME RELAY INTERCONNECTION

The first type of interconnection outlined in the FRF5 document is called Network Interworking [2] and corresponds to the case of an ATM network used as backbone between two Frame Relay networks.

An example of such a type of interconnection is given in figure 1.

The important features of the Network Interworking are:

1. Frame Relay traffic is transported across an ATM network such that one ATM Permanent Virtual Connection (PVC) is used to transport one or several Frame Relay PVCs. Thus, the mapping from ATM PVC to Frame Relay PVC can be *one to one*, or *one to many*.
2. As of present no interworking at the Switched Virtual Channel level has been defined yet, but current work is done in order to address this issue.
3. Rules, to map the congestion indication and the data discard in case of congestion, from the Frame Relay to the ATM layer, are used in order to guarantee the QoS levels.

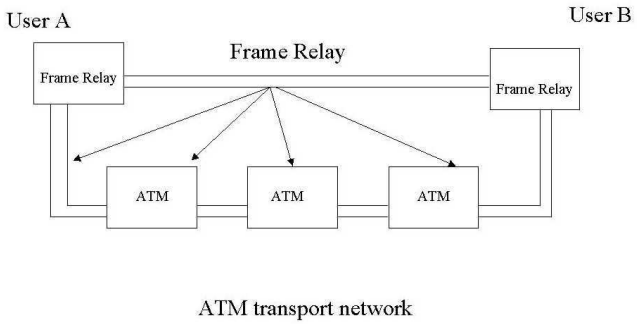

Figure 1: Network Interworking

The second type of interconnection is the Service Interworking defined in the FRF8 [3] document. A typical example of such a scenario is given in figure 2.
The goal in such an interconnection is to achieve the interoperability at a service level. A Frame Relay user directly inter-works with an ATM user located such that:

1. Each user is not "aware" that the other party uses a different technology.
2. Similarly to the previous case, no interworking at SVC level has been defined yet.
3. Depending on the transported information, some mapping might need to be performed. Such is the case for instance of IP traffic encapsulation, which is done differently for ATM and Frame Relay.

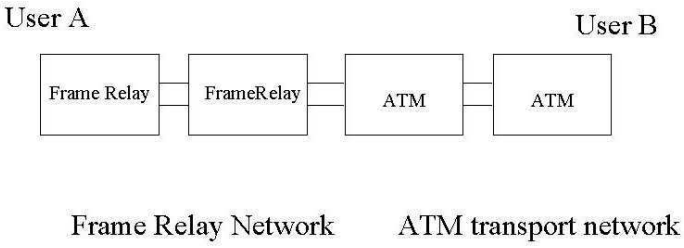

Figure 2: Service Interworking

2.2 AN EXTENDED INFORMATION MODEL

The main goal of this section is to present a network information model used to model ATM, Frame Relay networks and the interconnection between them. Previous work on building network models for the integrated management of ATM/SDH/WDM network was published in [4]. In this section we will extend the

model introduced in [4] in order to be used in the case of ATM/Frame Relay interconnections.

Before starting the presentation of this model, let us briefly review current and performed work on modeling transport networks. Briefly stated a network information model consists of abstractions of the actual network components. Its use is important for the delivery of services and to assure a proper interaction among applications, user and managed resources.

Our starting point was the model introduced in [4], which is closely related to the NRIM [5] model. The power of expression correlated with its simplicity and versatility were very promising for its potential reuse in the context of ATM and Frame Relay interconnection. The model itself is general allowing us to model both the ATM and the Frame Relay network.

By specializing the generic classes used to model the network resources we succeed to have an integrated model for a network based on both technologies. As a remark, the case of ATM and Frame Relay interconnection is particular suited to such an approach since both technologies are connection oriented sharing a lot of similarities. We did this specialization using the M4 document in order to provide the necessary parameters for the ATM part. The parameters for the Frame Relay part were obtained from the Frame Relay MIB, defined by the IETF.

An UML (Unified Modeling Language) diagram representing our model is given in figure 3.

Network level resources are modeled by several object classes. Relationships among these resources are represented by links. A link represents a semantic connection indicating an existent association between the involved resources. Role names and multiplicity indicators are further used to specify such an association. A multiplicity indicator defines the number of objects associated one to another.

Some of the information objects used in the model are:

SubNetwork. A subnetwork can be viewed at the lowest level to be the physical ATM switch. Several such switches connected by physical links can be regrouped to form another subnetwork and so on. Such a recursive division allows us to view an entire ATM network as a global switch. Allowing for several resolutions is of particular interest when one view is used by a customer, and the other one, much more accurate, showing the interior of an ATM network is used by the network operator in charge of managing the transport network.

LayerNetwork. A Frame Relay network, or an ATM network are modeled by such an entity. It represents a network using a characteristic transport technology.

Link. A physical link connecting two networks is a typical example for such an object. When a network is build of several layers, a link object at one layer can be supported by resources from lower level layers. A standard example is the case of an ATM/SDH network, where an ATM link is in fact built from several objects in the SDH network. In our case, if we consider the case of an Frame Relay Network Interworking scenario, a link in the Frame Relay layer is actually supported by one or several trails in the ATM network.

It is important to note that we generalized the relationship introduced in [4] where one link was supported by a trail object. In our model we can decompose a link into several interconnected trails. This proves to be useful in the case of a service interworking scenario in which a service trail used to interconnect a Frame Relay user to an ATM one, is in fact build from several layertrail objects. For this purpose we used a **Peer2Peer** class to model the Peer2Peer relationship introduced in the NRIM model. This particular feature is also one major reason why we considered a NRIM based model, like the one introduced in [4] as a starting point for our model.
The attributes of the Peer2Peer class were derived from the Service Interworking MIB [6], which is a IETF draft.

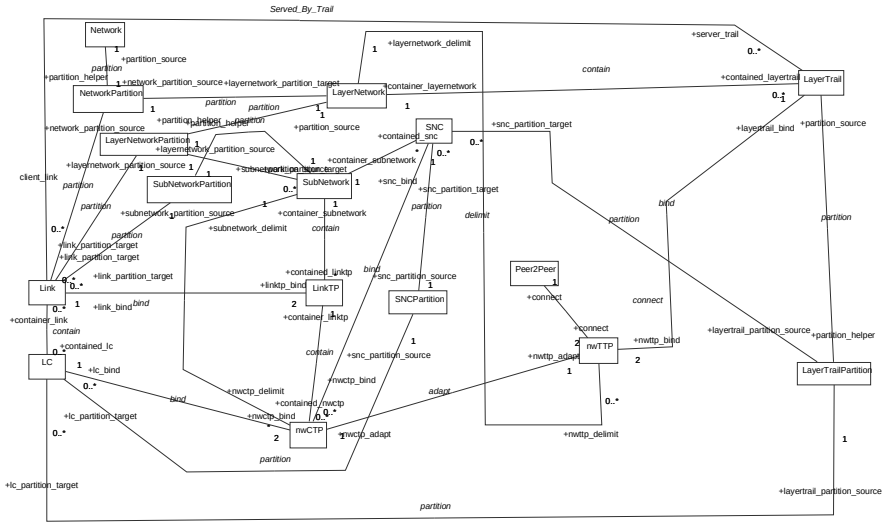

Figure 3: The network Information Model

LayerTrail. Is a resource transferring information needed in order to support a client link. It is important to note that a **LayerTrail** is constrained to be included in one **LayerNetwork**.

LC (Link Connection) is a resource transferring information across a link. In ATM for instance, a Virtual Path Link (the segment of a Virtual Path Link between two switches) is a typical example.

SNC (Subnetwork connections) are resources used to transfer information across a subnetwork. A subnetwork contains thus several **SNC**s. At the finest level of resolution, a SNC is a cross-connect in a switch.

Termination points delimit the two ends of a **LC** object, **Link** object or **LayerTrail** object. In case of a Service Interworking scenario, the interworking performed between two trails is modeled using a **Peer2Peer** object which is in a *connect* relationship with both corresponding endpoints. The parameters used to model this interconnections were derived from the current IETF draft regarding the service interworking MIB [6].

The termination points of a **LayerTrail** object can be associated to the termination points of a LC object using the *adapt* relationship . This is the case for instance, if a Link object is supported a **LayerTrai**l object. In this case the *adapt* relationships exists for every **nwTTP** (network Trail Termination Point) and all **nwCTP** (network Connection Termination Point) that are used by **LC** objects supported on that particular **Link** object.

In the networking interworking scenario between Frame Relay and ATM, this relationship is used in order to model the multiplexing of one or several Frame Relay PVC on one ATM PVC.

3. JAVA BASED IMPLEMENTATION OF THE NETWORK MODEL.

3.1 ARCHITECTURE

The major goal we are aiming at, is the realization of a software tool based on Java, using the proposed network model to manage a real world ATM/Frame Relay network. The major requirements we set on our developments were:

- The information model must be designed in the UML notation.
- The Java skeletons must be generated from a UML toolkit (here Rational ROSE).
- The server which implements those MOs must offer two interfaces: one
- application specific RMI interface and one OSI CMIS compliant view of the Managed Objects.

Based on these requirements, we have developed a Managed Object Server whose architecture is illustrated on figure 4.

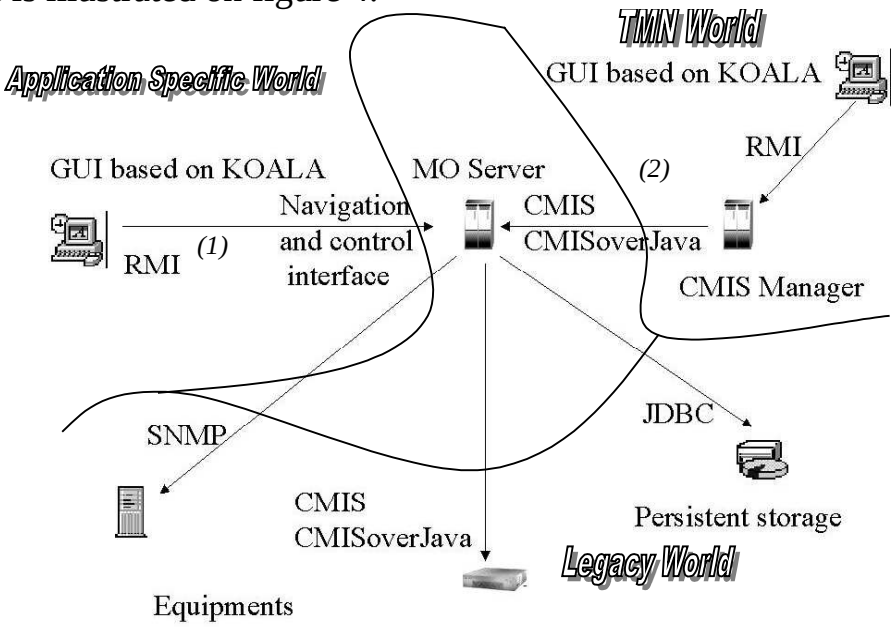

Figure 4. The Java management architecture

The core of the architecture is made of the Network Model Server. This server implements all managed objects representing the network. It accesses to equipment based management information via standard Java management service APIs (e.g. the Advent Java SNMP API or CMISoverJava [7] by our research group).

The MO Server exhibits 2 interfaces: one RMI interface with an application specific Java API and one CMIS interface through COJ. The first one offers an interface for enabling easy hierarchical navigation into the MOs. The hierarchy is based on the client/server relationships into a layer network. This interface offers a large set of methods for accessing configuration MOs and performing operations on these objects that are tuned towards the application defined in the next section.

Whereas the first interface could use the generated Java skeletons directly without any changes, the OSI one was built in an automated fashion by requiring each Java MO to implement a set of predefined interfaces making these objects OSI compatible. Those interfaces are:

- a naming interface which forces each MO to be able to report its OSI Distinguish Name (in fact its Relative Distinguish Name and a reference to its father object)
- a registration interface which enables each class and attribute to know its associated Object Identifier
- an ASN.1 compatibility Interface which forces each attribute to be able to represent any value of its type in the corresponding ASN.1 type.

In the first release of the prototype the ASN.1/GDMO specifications are built by hand from the UML design since it is required that containment relationships must be identified by the designer of the model in order to build the corresponding name-bindings and containment tree. Also, no automated mapping to ASN.1 from basic Java types are yet available.

3.2 THE CONFIGURATION APPLICATION

The tool that we developed is capable to offer a network view based on the above mentioned model to its user.
The model navigation permits several features among which we mention:

- Check for every component, all components which are used to support this object. For instance by clicking on a **Link** object we can visualize all subnetwork connections, **LCs** and Trails needed in the model to support this object. This dependence is checked in a recursive manner. For instance, assume that a **Link** object is supported by a **LayerTrail**. Next, we can further decompose this **LayerTrail** into **LC** objects and **SNCs**. One **Link** object containing one of the **LC** objects, is on its turn supported in a lower level layer. In this case we are able to navigate the model as deep as possible and to determine all these entities.

- Check for every component, all objects that are affected in case of a failure of this particular component. Thus we are able to see that for instance a failure or a misconfiguration of a subnetwork connection in the ATM layer will affect a particular **LC** object in the Frame Relay layer. Such a feature is useful in case of error correlation. If a management application is aware of several malfunctioning network entities, which might in fact come from a cascaded error propagation, the common root can be traced back, by navigating the object model (which is almost a tree, if some relationships that are not important for this purpose are ignored) and detecting the initial root fault.

A screenshot of an example case, where three ATM subnetworks are used to interconnect two Frame Relay networks is shown in figure 5.

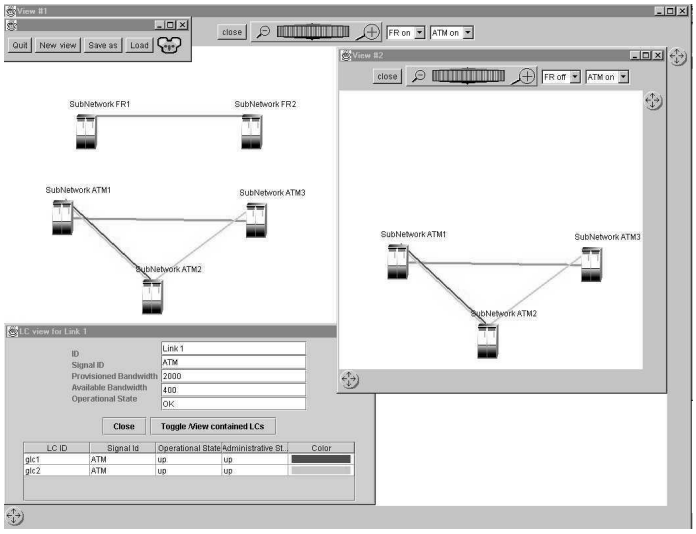

Figure 5. A screenshot from the implementation

This application has been built on top of each interface. The RMI one was much easier to build since the interface to the model was tuned for this application. On top of a CMIS stack, it took us much more time to build the requests necessary to rebuild all the relationships among MOs.

4. RELATED WORK

4.1. NETWORK INFORMATION MODELS

Several network information models have been already developed. Two major approaches can be pointed out. The first one, used especially in the TMN related groups focussed on telecommunication carrier networks and is summarized in the following.

The ATM community released the M4 network MIB as a standard in the ATM Forum. This document is related to modeling ATM networks and their characteristics. Related work, concerning the modeling of a generic transport

network has been done in the IUT community and were released in the M.3100 document [8] and G.805 document [9].

The TINA-C consortium also released a network model in the NRIM document [5], going even further and approaching also the service management task in a telecommunication network.

Finally, the European Telecommunication Standard Institute developed the GOM model [10].

Most of the above mentioned models have a large amount of overlap sharing major features. A network is represented in an object oriented way, where classes of objects characterize particular network components. Modeling is done in an independent way of the used underlying transport technology. Using techniques from the object oriented programming community, one can apply the general model to model particular networks by specializing the defined classes.

The semantic connections existing in a network are captured using relationships among the existing classes.

The second approach in building network information models came from the IP community and is endorsed by groups like the Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF)[11].

4.2. JAVA BASED MANAGEMENT

The use of the Java technology in management has been proposed in many contributions within the last four years. Today, several Java-based architectures are proposed for management integration [12],[13],[14],[15] and [16]. These proposals are in fact very attractive and represent undoubtedly the framework for tomorrow's Java-based Management. These architectures do unfortunately not meet yet the requirements expressed in this project. None of them offers direct Java-based access to TMN-based Management Systems nor do they offer an OSI Agent-interface to upper layer management solutions. These drawbacks can be solved in using the J^{TMN} [17] and COJ environments developed in our group.

5. CONCLUSIONS AND FUTURE WORK.

In this paper we introduced a network level information model for the representation of ATM and Frame Relay networks. The model extends the work introduced in [4] which was developed in the context of integrated management of ATM/SDH/WDM networks. The extension is due to the service interworking model allowed in ATM/FR interconnection requiring the introduction of a peer-to-peer relation between endpoints of different layer networks.

A Java based implementation of this model has been successfully realized, demonstrating the viability and usability of the model. The implemented architecture illustrates especially that one can easily combine the use of UML development tools and implement the system in Java so that multiple views of its contained managed objects (i.e. a standard CMIS interface as well as a complete application specific RMI interface), are offered. Here we have shown that a UML design of the information model can serve both pure Java implementation of a MO

server as it is done in JMAPI for example and for CMIS compliant interfaces by making an explicit separation between the MOs and the external hierarchical view and naming issues.
Future work will consist in the integration with a real network management platform for an ATM/Frame Relay network with the emergence of standard Java API's for management. Especially the use of JMX standard interfaces as well as the Marvel environment by Anerousis et Al. [12] will be explored.

REFERENCES

[1] T. Quatrani, *Visual Modelling with Rational Rose and UML*: Addison-Wesley, 1998.
[2] D. O'Leary, "Frame Relay/ ATM PVC Network Interworking Implementation Agreement (FRF5)," : Frame Relay Forum, 1994.
[3] D. O'Leary, "Frame Relay/ ATM PVC Service Interworking Implementation Agreement (FRF8)," : Frame Relay Forum, 1995.
[4] C.-C. Shen and J. Y. Wei, "Network-Level Information Models for Integrated ATM/SONET/WDM Management," presented at IEEE/IFIP 1998 Network Operations and Management Symposium, New Orleans, USA, 1998.
[5] "Network Resource Information Model," : TINA-C, 1997.
[6] K. Rehben, O. Nicklass, and G. Mouradian, "FR/ATM Service Interworking MIB," : IETF, 1999.
[7] L. Andrey, O. Festor, and R. State, "COJ: A free CMIS Compliant Java API and its Various Implementations," , 2000.
[8] "Generic Network Information Model : Recommendation 3100," : CCITT, 1992.
[9] "G.805-Recommandation UIT-T G.805: Architecture fonctionelle générale des réseaux de transport," : ITU-T, 1995.
[10] "Generic managed object class library for the network level view," : ETSI, 1996.
[11] DMTF, "The Common Information Model," .
[12] N. Anerousis, "A Distributed Computing Environment for Building Scalable Management Services," presented at IFIP IEEE IM'99, Boston, 1999.
[13] F. Barillaud, L. Deri, and M. Feridun, "Network Management using Internet Technologies," presented at IFIP IEEE IM'97, San Diego, CA, 1997.
[14] SUN, "The Java-Management API," . http://java.sun.com/products/JavaManagement/, 1999.
[15] J. Horowitz, "Bringing Java to TMN," presented at NOMS'98, New Orleans, 1998.
[16] H. Dassow, C. Hubert, B. Frohnhoff, and G. Aschemann, "Realization of a TMN Java Management API," presented at NOMS'98, New Orleans, 1998.
[17] O. Festor, P. Festor, L. Andrey, and N. B. Youssef, "JTMN: A full Java-based TMN Development and Experimentation Environment," , 2000.